

Identity for getUserMedia() + Recording Consent Issues

IETF 84.5

Eric Rescorla

ekr@rtfm.com

Background

- The identity mechanism is intended to allow authentication that isn't provided by the calling site
- For high security, we want the site to be able to neither see nor touch media

The API MUST provide a mechanism for the requesting JS to relinquish the ability to see or modify the media (e.g., via `MediaStream.record()`). Combined with secure authentication of the communicating peer, this allows a user to be sure that the calling site is not accessing or modifying their conversation.

— draft-ietf-rtcweb-security-arch-05

- This is not required for all applications but needs to be available

General Idea: Grant Media Access to an Identity

- In basic gUM, user gives permission to *site* to access devices
- What we want is to give permission to *identity* to access devices
 - Site just manipulates handles to the streams
 - But can't access them

Topics

- How do we isolate the stream? And what can you do with an isolated stream?
- How does the user/browser learn the identity?
- How and when does the user give consent and to what?

Stream Isolation

- How do you isolate a stream?
 - Tag it as being in a separate origin
 - Normal cross-origin protections apply
 - This is how cross-origin video streams work *now*
- What can you do with it?
 - Attach to a video/audio element
 - * But that element can't be modified/read from content
 - `pc.addStream()`*

*Subject to permissions. See later slides

History

- Discussed this in Lyon
- A bunch of syntactic proposals
- Martin and I proposed a general direction people seemed to like
 - Which I then never got around to writing up
- Here is a detailed description

Three ways to call gUM (all using constraints)

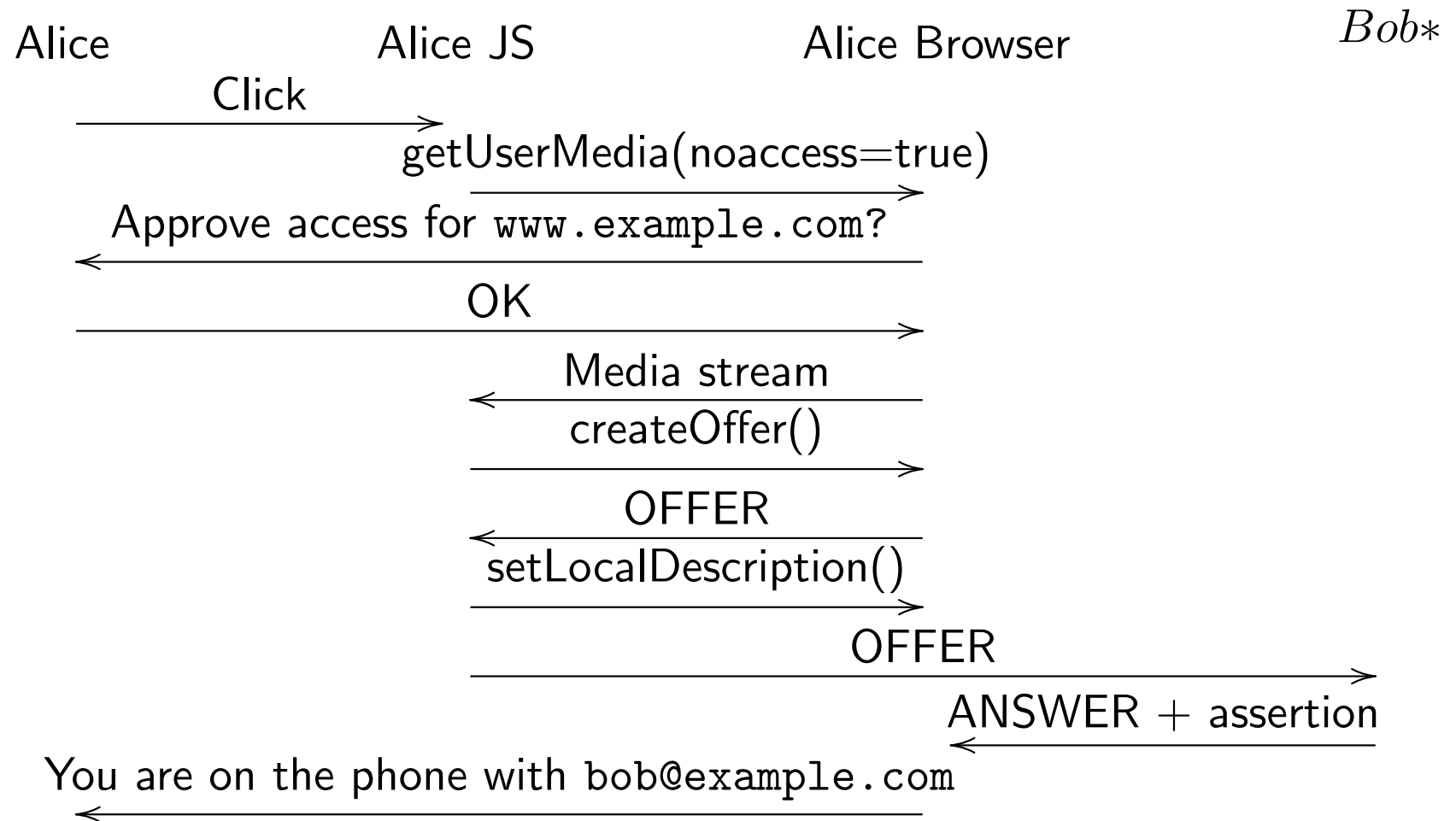
- Call `getUserMedia()`
 - Current behavior
- Call `getUserMedia(noaccess=true)`
 - gUM permissions check done based on origin
 - Media inaccessible to site
- Call `getUserMedia(peerIdentity=bob@example.com)`
 - gUM permissions check done based on identity/origin pair
 - Media inaccessible to site

Call `getUserMedia()` as normal

- Current behavior

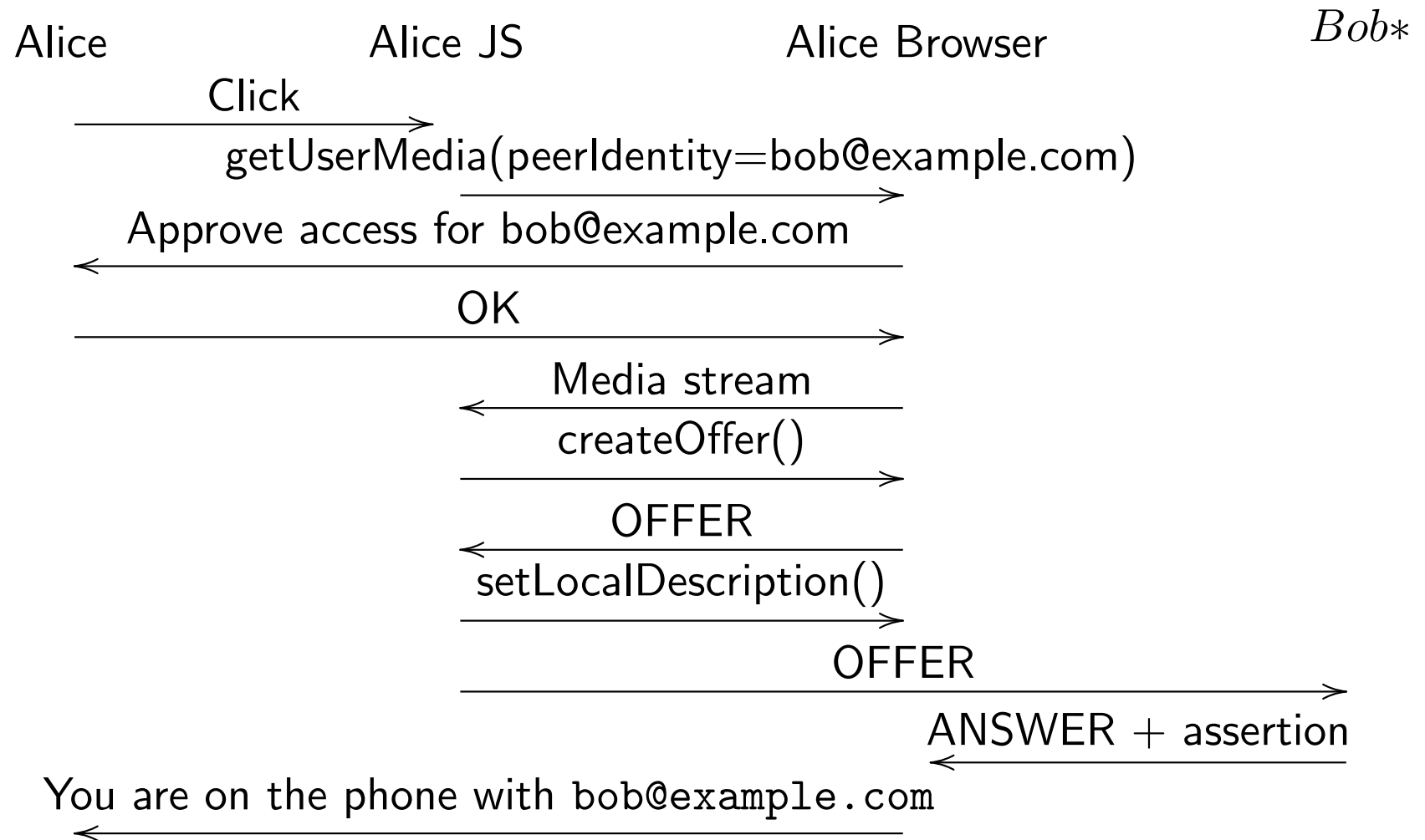
`noaccess=true`

- Media permissions checks behave as usual
- Stream is isolated
- Keying must be via DTLS
- Once PeerConnection is established, browser knows peer identity
 - Browser displays identity and a “no access” indicator in the UI



`peerIdentity=bob@example.com`

- Browser prompts user with “do you want to share your camera with 'foo@example.com'”
- (Or “Bob Jones” if he is in your address book)
 - Not the site’s address book
- Stream is isolated
- Keying must be via DTLS
- `MediaStream` can only be connected to a PC with identity `bob@example.com`



Recording Consent

- W3C is publishing recording API as FPWD
- What, if any, are the implications for consent?
- In particular, do we need to ask for “recording consent”?

Basic Recording Concept (review)

- Create a `MediaStream` in the usual way
- Feed it to the `MediaRecording` constructor
- This gives direct access in the JS to the media

What's the argument for special permissions?

“We tackled the recording issue at TokBox about a year ago. We found that 11 states require all parties to know and consent to being recorded, which is different from consenting to use the camera for a live conversation. The safest approach is to have a pop-up that specifically requests permissions for recording. I think allowing developers to freely record without consent will lead to legal problems for well intentioned developers who are unaware of the laws they are dealing with. Asking consent for the camera, but not to record, seems to imply that recording is acceptable when that may not be the case.

With regards to when to ask permission, we found it was better to ask before a call was started because it was disruptive to pop up a consent dialog during the flow of conversation. If an application transitions to start recording at the same time it connects two people in a call, you may miss the beginning of the conversation while both people are dealing with the consent dialogs, and it may complicate syncing multiple recordings if they start at wildly different times. You may also only want to allow people to use the application if they are willing to be recorded.”

– Ben Pedrick

Not the only way to get direct access

- Pipe MediaStream to a PC
 - Record on server
 - Maybe send raw bits back to client (if needed)
- Video can be read off a canvas
- No technical measure can distinguish recording from a WebRTC call to an arbitrary location

Proposed Resolution

- Permissions check only on getUserMedia
- Recording treated like data to an arbitrary (non-identity PC)
- Application's job to inform the user of when recording is taking place